

FIG. 1

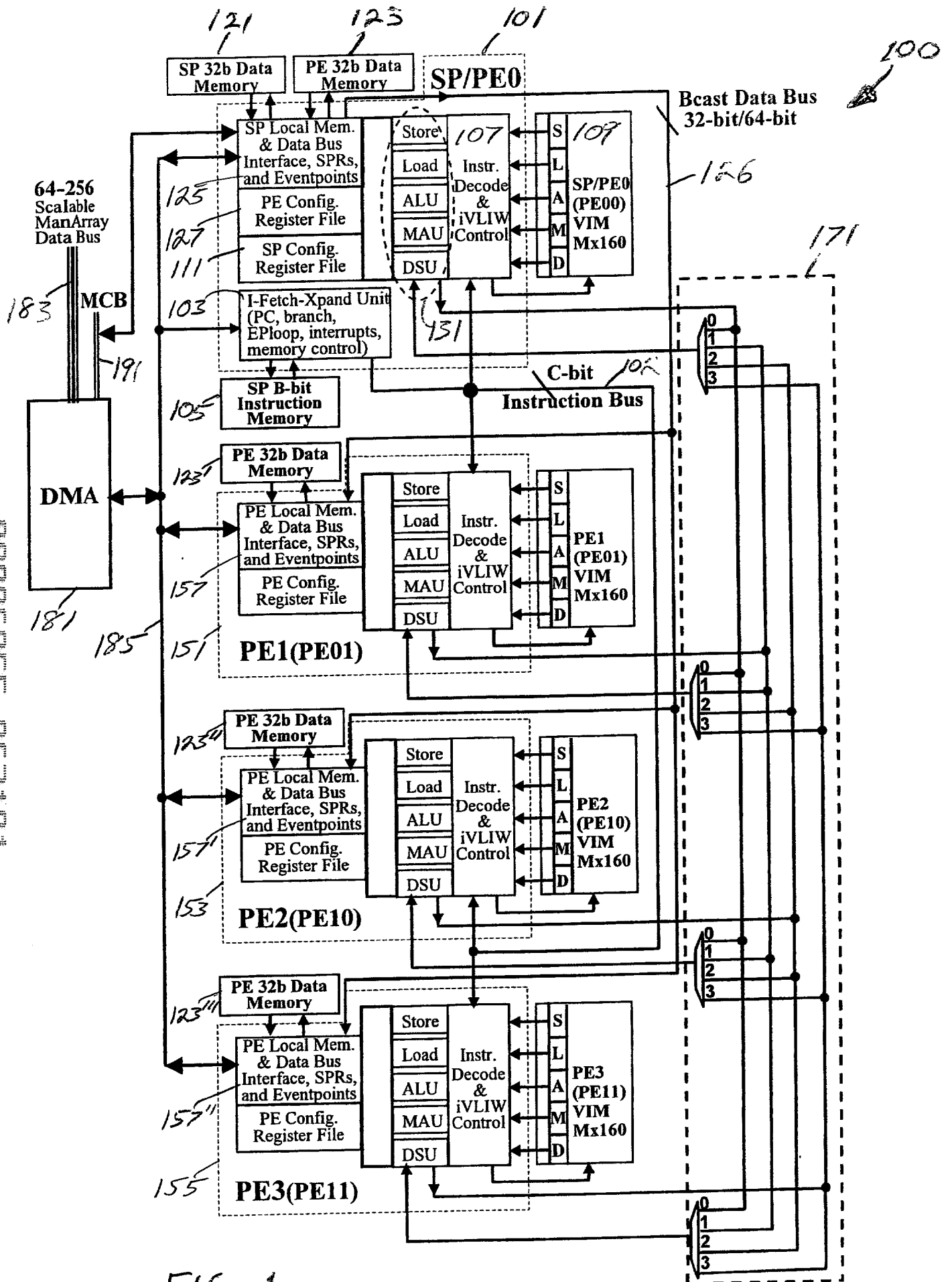


FIG. 1

LV Encoding

200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Group		S/P		CtrlOp			E/D		UAF		InstrCnt			0	0	0	SU	LU	ALU	MAU	DSU	Vb	0	VIMOFFS							

FIG. 2A

LV Syntax/Operation

210

Instruction	Operands	Operation
LV.[SP]	V[01], VIMOFFS, InstrCnt, D={SLAMD}, F={AMDN}	<p>(V[01]+VIMOFFS)[SU].enable ← 0 if (D=S) (V[01]+VIMOFFS)[LU].enable ← 0 if (D=L) (V[01]+VIMOFFS)[ALU].enable ← 0 if (D=A) (V[01]+VIMOFFS)[MAU].enable ← 0 if (D=M) (V[01]+VIMOFFS)[DSU].enable ← 0 if (D=D)</p> <p>(V[01]+VIMOFFS)[UAF] ← ALU if (F=A or F=) (V[01]+VIMOFFS)[UAF] ← MAU if (F=M) (V[01]+VIMOFFS)[UAF] ← DSU if (F=D) (V[01]+VIMOFFS)[UAF] ← None if (F=N)</p> <p>for (i=0; i&lt; InstrCnt; i++) {   Load instruction into (V[01]+VIMOFFS)   if (SU Instr AND D != S) { (V[01]+VIMOFFS)[SU].enable ← 1 }   if (LU Instr AND D != L) { (V[01]+VIMOFFS)[LU].enable ← 1 }   if (ALU Instr AND D != A) { (V[01]+VIMOFFS)[ALU].enable ← 1 }   if (MAU Instr AND D != M) { (V[01]+VIMOFFS)[MAU].enable ← 1 }   if (DSU Instr AND D != D) { (V[01]+VIMOFFS)[DSU].enable ← 1 } }</p>

FIG. 2B

FIG. 2A

XV Encoding

300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Group		S/P		CtrlOp			VX	UAF		0	0	0	0	0	0	0	SU	LU	ALU	MAU	DSU	Vb	0	VimOffs							

FIG. 3A

XV Syntax/Operation

310

Instruction	Operands	Operation
XV.[SP]	V[01], VIMOFFS, E={SLAMD}, F={AMDN}	Execute (V[01]+VIMOFFS)[SU] if (E=S) Execute (V[01]+VIMOFFS)[LU] if (E=L) Execute (V[01]+VIMOFFS)[ALU] if (E=A) Execute (V[01]+VIMOFFS)[MAU] if (E=M) Execute (V[01]+VIMOFFS)[DSU] if (E=D)  (V[01]+VIMOFFS)[UAF] ← ALU if (F=A) (V[01]+VIMOFFS)[UAF] ← MAU if (F=M) (V[01]+VIMOFFS)[UAF] ← DSU if (F=D) (V[01]+VIMOFFS)[UAF] ← None if (F=N)

FIG. 3B

FIG. 3A

400  
functionA:

402 — lv.p v0, 0, 2      ! load VLIW 0 with the next 2 instructions  
      instr1 } 404  
      instr2 }

412 — lv.p v0, 1, 3      ! load VLIW 1 with the next 3 instructions  
      instr3 } 414  
      instr4 }  
      instr5 }

420 — xv.p v0,0,e=AM      ! execute VLIW 0, enabling units A and M  
422 — xv.p v0,1,e=AMS      ! execute VLIW 1, enabling units A, M and S

ret

FIG. 4A

function A':

xv.p v0,0,e=AM      ! execute VLIW 0, enabling units A and M  
xv.p v0,1,e=AMS      ! execute VLIW 1, enabling units A, M and S

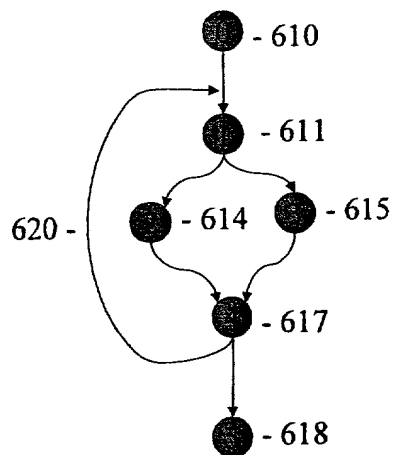
ret

FIG. 4B

500

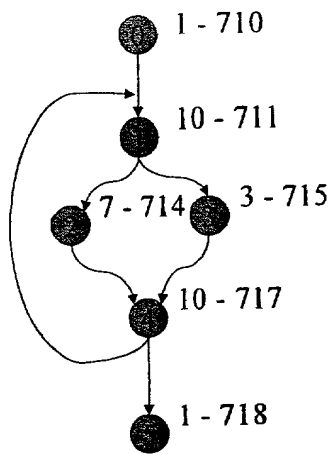
510 - 0: Program start  
511 - 1: loop 10 times  
512 -     execute VLIW a  
513 -     if condition then  
514 - 2:     execute VLIW b  
515 - 3:     else  
516 -     execute VLIW c  
517 -     end if  
518 - 4: end loop  
518 - 5: Program end

FIG. 5



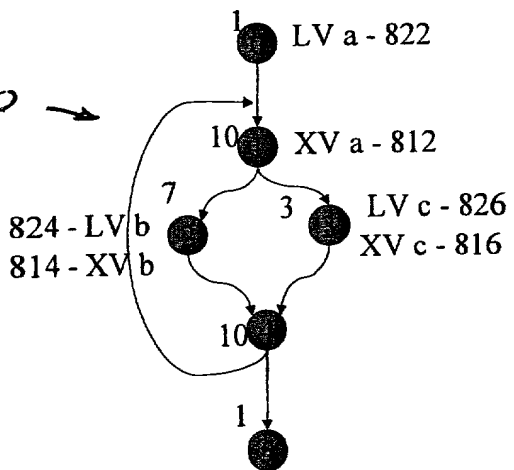
600

FIG. 6



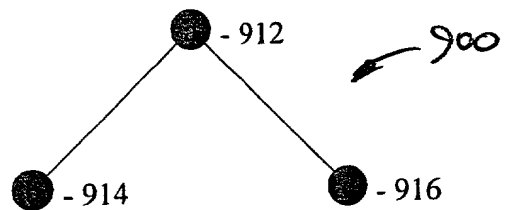
700

FIG. 7



800

FIG. 8



900

FIG. 9

1000

```
1001 - done := false;
1002 - while not done do {
1003 -     done := true;
1004 -     BestImprovement := 0;
1005 -     for each Lvi from LVlist do {
1006 -         [NewState, improvement] := MoveUp(Lvi, CurrentState);
1007 -         if improvement > BestImprovement then {
1008 -             BestState := NewState;
1009 -             BestImprovement := improvement;
1010 -             done := false;
        }
    }
}
1011 - if not done then {
1012 -     CurrentState := BestState;
}
}
```

FIG. 10